# APPLICATION

FOR

# UNITED STATES LETTERS PATENT

APPLICANT NAME R. G. Hartmann, et al.

TITLE SYSTEM AND METHOD FOR SERVER DISPLAY CONFIRMATION RECORD RESPONSE IN A CONNECTION ORIENTED CLIENT/SERVER PROTOCOL

DOCKET NO. END920010020US1

**INTERNATIONAL BUSINESS MACHINES CORPORATION**

CERTIFICATE OF MAILING UNDER 37 CFR 1.10

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Commissioner of Patents and Trademarks, Washington, D.C., 20231 as "Express Mail Post Office to Addressee" on 8-17-01

Mailing Label No. EL598672687US

Name of person mailing paper: Georgia Y. Brundege

Georgia Y. Brundege 8-17-01

Signature Date

a:formal2.lwp

# SYSTEM AND METHOD FOR SERVER DISPLAY CONFIRMATION RECORD RESPONSE IN A CONNECTION ORIENTED CLIENT/SERVER PROTOCOL

## Cross Reference to Related Application

This application is a Continuation-In-Part of U.S. Patent application S/N 09/827,012, filed 5 April 2001 by R. G. Hartmann, et al. for System and Method for Server display Confirmation Record Response in a Connection Oriented Client/Server Protocol.

## Background of the Invention

Technical Field of the Invention

This invention pertains to connection oriented client/server negotiation protocols. More specifically, it pertains to Telnet negotiation protocols for display and printer sessions allowing transfer of default or defined custom information within a confirmation record at the request of the client.

Background Art

There is a need in the art to enable a Telnet client when attempting to connect to a Telnet server to obtain connection status information including, for example, why did a connection request fail; why did a client auto-sign-on request fail; or what is the name of the virtual terminal display device assigned to this client. Auto-sign-on requests may fail, for example, because of an incorrect password or profile, a disabled or unknown profile, required encryption, expired user, and so forth.

This traditional Telnet support is accomplished in accordance with the following suite of Network Working Group Request for Comments (RFCs): Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, May 1983; Postel, J. and J. Reynolds, "Telnet Option Specifications", STD 8, RFC 855, May 1983; Postel, J. and J. Reynolds, "Telnet Binary Transmission", STD 27, RFC 856, May 1983; VanBokkeln, J., "Telnet Terminal-Type Option", RFC 1091, February 1989; Postel, J. and J. Reynolds, "Telnet End of Record Option", RFC 885, December 1983; Alexander, S., "Telnet Environment Option", RFC 1572, January 1994; Chmielewski, P., "5250 Telnet Interface", RFC 1205, February

1991; Postel, J. and J. Reynolds, "Telnet Supress Go Ahead Option", STD 29, RFC 858, May 1983; and Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

The above suite of referenced RFCs jointly and severely fall short of providing an understanding of why a connection request has failed, and such is needed in the art to enable a client to correct the problem and retry a connection request such that it will be successful.

Similarly, when a connection request has succeeded, the client may need to know additional information, such as the name of the virtual terminal display device assigned to this client. Knowing the device name of a client connection is useful for audit logging, billing and error analysis for connected clients. Heretofore, screen scraping technology has been employed to acquire such a device name, relying on the screen layout to analyze the location of the device name on the screen. If the sign-on panel is altered such that the device name is in a different location, screen scraping fails. Also, this screen scraping technology does not work when the sign-on panel is bypassed.

In a client/server network, both client terminal and printer emulators often connect to a server on a host system. This host system can do different kinds of processing on the client session request based on client information, such as IP address, terminal or printer device requested, and auto-signon information. Some of the things that can be done include: accept or deny the connection based on the IP address or port; allow or disallow access to the request display or printer device for authority reasons, or switch the name of the requested device; route the client session to a particular sub-system on the host for processing, such as for workload balancing or language support; perform auto-signon services for the client, bypassing the logon screen; perform audit and security logging on the connection request; associate a client session with other client sessions running on the host, such as associating printer with a display session; and run a custom exit program to do "anything" the system owner desires.

It is often desirable to return the result of this processing to the client emulator. The client emulator can take advantage of information in various ways. Some of these include: post the name of the assigned terminal or

printer device that was allocated by the host; post the name of an associated terminal or printer device that was linked to this session by the host; when printing, the client will know where (which printer, what building, what room) output can be picked up; tell the client what kind of security level the system is running, and which kind of password encryption is required; if a particular request, such as for device name, is rejected, retry with another device; if auto-signon is failing, client would like some indication why, such as password expired, profile disabled, no such profile, system lockout -- so the problem can be automatically fixed; if system is overloaded, client would like to know session connection request was denied for workload reasons, such as off-peak hours, to try later or be redirected to another host; and read and interpret any custom information sent by the server side exit program.

These are only some of the possible things a client emulator could exploit, and there are many custom applications that could be done simply if both the server and the client could run exit programs on the session connection request. However, none of this can be done without a mechanism to return the results of this processing from the server to the client emulator. There is a need,

therefore, in the art to allow client emulators to request that custom information be returned by the server thereby allowing the customers to exploit custom solutions between clients and servers.

It is, therefore, an object of the invention to provide an improved system and method for client/server session connection.

It is an object of the invention to provide an improved system and method for establishing a client/server connection.

It is a further object of the invention to provide an improved system and method for negotiating a client/server connection in a connection-oriented protocol.

It is a further object of the invention to provide a system and method allowing customers to exploit custom solutions between clients and servers.

It is a further object of the invention to provide a system and method for enabling client emulators to request that custom information be returned by servers.

It is a further object of the invention to provide a system and method for exploiting confirmation records technology to enable clients to receive custom information from servers during session connection.

## Summary of the Invention

A system and method for operating a client and a server to establish a network connection.

A system and method for operating a client and a server to establish a network connection. Environment parameters are negotiated for establishing a connection-oriented connection of the client to the server, the parameters optionally including an invitation on the part of the server for the client to request a custom confirmation record. Responsive to that invitation, the client may request a default custom confirmation record or a defined custom confirmation record.

In accordance with an aspect of the invention, there is provided a computer program product configured to be operable to operating a server in a network according to method steps including providing to a client an optional default or defined custom confirmation record.

Other features and advantages of this invention will become apparent from the following detailed description of the presently preferred embodiment of the invention, taken in conjunction with the accompanying drawings.

## Brief Description of the Drawings

Figure 1 is a system diagram illustrating a client/server system.

Figure 2 is a diagram illustrating the format of a response record in accordance with the preferred embodiment of the invention.

Figure 3 is a flow chart representation of negotiations for a default custom confirmation record in accordance with the preferred embodiment of the invention.

## Best Mode for Carrying Out the Invention

Parent application, Hartmann, et al. S/N 09/827,012, (hereafter, Hartmann) supra, relates to a confirmation record requested by a client from a server. The server responds in the confirmation record with a return code that indicates the success or reason for failure to satisfy the request, and allows for data to be exchanged only in the direction from server to client.

The present invention provides a customized confirmation record. The information is exchanged in the same confirmation record as in the parent application, however, the client can request of the server application additional predefined custom information.

Referring to Figure 1, a confirmation record technology is provided for connection oriented client/server sessions, such as TCP/IP Telnet display sessions.

A typical protocol stack includes application 30, transport 32, network 34 and link layers 36. Telnet, for example, is an application that executes in application layer 30 and, as is represented by line 46, is in virtual

connection or communication with application layer 31 at server 42. In such a protocol stack, communication is between corresponding layers. Thus, application layer may be in communication with application layer 30, transport layer 32 with layer 33, network layer 34 with network layer 35, and link layer 36 is physically connected through network 44 with link layer 37.

This confirmation record technology is described hereafter and in T. Murphy, Jr., P. Rieth, J. Stevens, "5250 Telnet Enhancements", Network Working Group Request for Comments (RFC): 2877, July 2000, the teachings of which are incorporated by reference. In RFC 2877, a version of a confirmation record is used by SNA printers. In this version, the confirmation record is always sent to printer emulators and returns the name of the host system, the name of the virtual printer device assigned and any error or success codes. With this technology, a Telnet client 40, for example, can connect to a Telnet server 42 over a network connection 44 and optionally request a detailed return code that describes the status of the connection. With the information of the return code, the client 40 is able to ascertain in the event of a successful connection the name of the virtual display device assigned to this

client 40, and in the event of an unsuccessful connection the information required to correct the problem and retry a connection request such that it is successful. In the event of a successful connection, the return code, or confirmation record, allows the client to know the virtual terminal device name without the need to employ a screen scrape scheme to analyze the sign-on panel, assuming it is even available. Knowing the virtual terminal device name enables the client to assign a session name to the GUI window for the client emulator. Also, knowing the device name of a client connection is very useful for audit logging, billing and error analysis for connection clients.

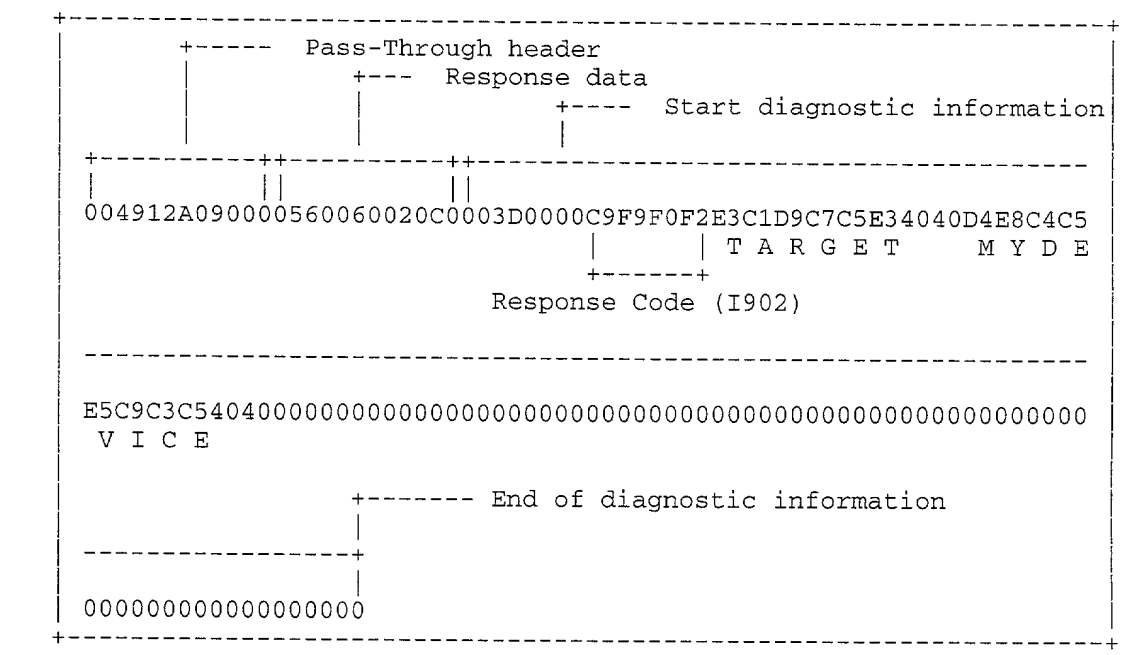
Referring to Figure 2, the format of a response record 100 includes pass through header 102, response data 104, and diagnostic information 106. Pass through header 102 includes length field 108, header 110, and several characters from fixed value fields 112. Response data 104 includes several characters from field 112. Diagnostic information includes a few characters from field 112, response code 114, system name 118 and device name 120. In accordance with the present invention, the format of the response record is extended to allow the insertion of application specific information that can be passed at the

application level between server and client systems. This new record layout 122 allows for the server 42 to return custom (default or defined) information 124 to the client 40, and for the client to react to information from the server. This application specific information can come from exit programs running in application layers 30, 31 on either or both the server and client machines, or can come from a custom version of either the server or client. In accordance with a preferred embodiment of the invention, this extension 124 to the confirmation record is optional, and must be specifically requested by the client emulator in order for it to be returned, thus insuring compatibility with emulators that do not support it. If custom confirmation record 122 is not specifically requested via the new environment variable, then the old version of the confirmation record 100 is sent.

Table 1 presents an example (from the parent application) of a success response record 100 according to the format of Figure 2, and Table 2 presents an error response record 100 according to the same format. Table 3 gives some of the response codes 114 for a success response 100 and Table 4 some of the response codes 114 for an error response record 100. The response record in Table 2 is one

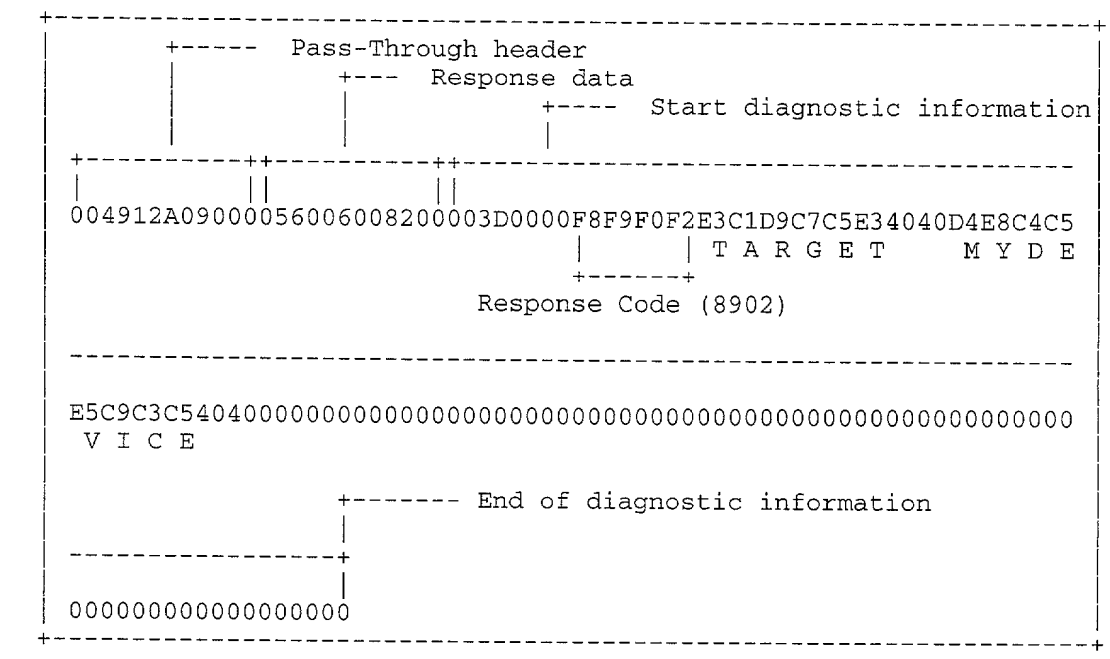
that reports an error. In this example , the virtual device named "MYDEVICE", is not available on the target system "TARGET", because the device is not available. This error may indicate that the device was already assigned to another Telnet session.

**Table 1 Example Success Response Record**




- '0049'X = Length pass-through data, including this length field
- '12A0'X = GDS LU6.2 header
- '90000560060020C0003D0000'X = Fixed value fields
- 'C9F9F0F2'X = Response Code (I902)
- 'E3C1D9C7C5E34040'X = System Name (TARGET)
- 'D4E8C4C5E5C9C3C54040'X = Object Name (MYDEVICE)

**Table 2 Example Error Response Record**




Example of an error response record.

- '0049'X = Length pass-through data, including this length field
- '12A0'X = GDS LU6.2 header
- '90000560060020C0003D0000'X = Fixed value fields
- 'F8F9F0F2'X = Response Code (8902)
- 'E3C1D9C7C5E34040'X = System Name (TARGET)
- 'D4E8C4C5E5C9C3C54040'X = Object Name (MYDEVICE)

**Table 3 Start-Up Response Record Success Response Codes**

| CODE | DESCRIPTION |
|---|---|
| I901 | Virtual device has less function than source device |
| I902 | Session successfully started |
| I906 | Automatic sign-on requested, but not allowed. Session still allowed; a sign-on screen will be coming. |

**Table 4 Start-Up Response Record Error Response Codes**

| CODE | DESCRIPTION |
|---|---|
| 2702 | Device description not found. |
| 2703 | Controller description not found. |
| 2777 | Damaged device description. |
| 8901 | Device not varied on. |
| 8902 | Device not available. |
| 8903 | Device not valid for session. |
| 8906 | Session initiation failed. |
| 8907 | Session failure. |
| 8910 | Controller not valid for session. |
| 8916 | No matching device found. |
| 8917 | Not authorized to object. |
| 8918 | Job canceled. |
| 8920 | Object partially damaged. |
| 8921 | Communications error. |
| 8922 | Negative response received. |
| 8923 | Start-up record built incorrectly. |
| 8925 | Creation of device failed. |
| 8928 | Change of device failed. |
| 8929 | Vary on or vary off failed. |
| 8930 | Message queue does not exist. |
| 8934 | Start-up for S/36 WSF received. |
| 8935 | Session rejected. |
| 8936 | Security failure on session attempt. |
| 8937 | Automatic sign-on rejected. |
| 8940 | Automatic configuration failed or not allowed. |
| I904 | Source system at incompatible release. |

Referring to Figure 3, method steps of an exemplary negotiation for a custom confirmation record in accordance with a preferred method of the invention are summarized.

In step 50, server 42 invites client 40 to engage in new environment negotiations. These negotiations are conducted in accordance with procedures described in S. Alexander, "Telnet Environment Options Negotiations", RFC 1572, Jan. 1994.

In step 52, client 40 accepts the invitation to negotiate a new environment.

In step 54, server 42 opens negotiations for terminal type, which client 40 accepts in step 56.

In step 70, server 42 instructs client 40 to send several parameters, and extends two invitations to the client to request additional information, including VAR and USERVAR. In step 72 client 40 responds. In accordance with the preferred embodiment of the invention, in the response of step 72, client 40 requests with the code "USERVAR 'IBMSENDCUSTOMCONFREC' VALUE 'YES'" that server 42 send a default custom confirmation record 122. Alternatively, such a request may be implied from some other parameter in connection with the new environment negotiations. Thus, for example, client 40 may have to specifically request a confirmation record 100 when requesting connection of a virtual display device, but such would be implied when requesting connection of a virtual printer device. Client 40 may also respond to the USERVAR invitation of step 70 with USERVAR "IBMSENDCUSTOMCONFREC" having a VALUE not "YES". In this case, the VALUE would specify a defined custom confirmation record in the form, for example, of a

space delimited list. In the case of such a defined custom confirmation record, server 42 could have been provided with an exit program for execution within application layer 31 for responding to each prospective item in the delimited list.

In a client/server network, both client terminal and printer emulators often connect to a server on a host system. This host system can do different kinds of processing on the client session request based on client information, such as IP address, terminal or printer device requested, and auto-signon information. Some of the things that can be done include: accept or deny the connection based on the IP address or port; allow or disallow access to the request display or printer device for authority reasons, or switch the name of the requested device; route the client session to a particular sub-system on the host for processing, such as for workload balancing or language support; perform auto-signon services for the client, bypassing the logon screen; perform audit and security logging on the connection request; associate a client session with other client sessions running on the host, such as associating printer with a display session; and run a custom exit program to do "anything" the system owner

desires.

In accordance with the present invention, a system and method is provided for returning the result of this processing to the client emulator in the form of various return codes or other information items. The client emulator can take advantage of information in various ways. Some of these include: post the name of the assigned terminal or printer device that was allocated by the host; post the name of an associated terminal or printer device that was linked to this session by the host; when printing, the client will know where (which printer, what building, what room) output can be picked up; tell the client what kind of security level the system is running, and which kind of password encryption is required; if a particular request, such as for device name, is rejected, retry with another device; if auto-signon is failing, client would like some indication why, such as password expired, profile disabled, no such profile, system lockout -- so the problem can be automatically fixed; if system is overloaded, client would like to know session connection request was denied for workload reasons, such as off-peak hours, to try later or be redirected to another host; and read and interpret any custom information sent by the server exit program.

A set of these above described information codes or items may be returned to a client in a default custom confirmation record (such as in response to IBMSENDCUSTOMCONFREC VALUE YES), and other items defined by way of personalized exit programs for return in a defined custom confirmation record (such as when IBMSENDCUSTOMCONFREC VALUE is not YES but rather a list of one or more specific information items.)

Negotiations continue in steps 76-92, for such additional negotiations as end-of-record and binary, and thereafter server 42 transmits the requested default or defined confirmation record, followed in case of a successful connection with the data stream.

In Table 5, an expanded example is presented of environment option negotiations in accordance with the parent application. As shown, clear text is followed by hex representation. Thus, line 2 'FFFD27' is the hex representation of line 1 'IAC DO NEW-ENVIRON', lines 13-14 are the hex representation of lines 9-12, and lines 58-62 are a hex representation of the confirmation record of Figure 2. The request for a confirmation record is illustrated at line 24. In line 59, the hex value

'C9F9F0F2' represents the successful return code 114 of I902 (see Table 3), and the device name 120 assigned to this virtual device is in the following ten hex bytes 'D1C5C6C6 E2C4E2D7 4040' on lines 59 and 60. IAC is a Telnet option negotiation code meaning "Interpret as command", SB represents "begin" and SE "end".

**Table 5: TN5250E Environment Option Negotiations**

| | Telnet Server | | Telnet Client |
|---|---|---|---|
| 1 | IAC DO NEW-ENVIRON | -> | |
| 2 | FFFD27 | | |
| 3 | | <- | IAC WILL NEW-ENVIRON |
| 4 | | | FFFB27 |
| 5 | IAC DO TERMTYPE | -> | |
| 6 | FFFD18 | | |
| 7 | | <- | IAC WILL TERMTYPE |
| 8 | | | FFFB18 |
| 9 | IAC SB NEW-ENVIRON SEND | | |
| 10 | USERVAR "IBMRSEEDxxxxxxxx" | | |
| 11 | USERVAR "IBMSUBSPW" | | |
| 12 | VAR USERVAR IAC SE | -> | |
| 13 | FFFA2701 0349424D 52534545 | | |
| 14 | 447D68B9 2BE04E04 040003FF F0 | | |
| 15 | | | IAC SB NEW-ENVIRON IS |
| 16 | | | VAR "USER" VALUE "JSTEVENS" |
| 17 | | | USERVAR "IBMRSEED" VALUE |
| 18 | | | USERVAR "IBMSUBSPW" VALUE |
| 19 | | | "YYYYYYYY" |
| 20 | | | USERVAR "DEVNAME" VALUE "JEFFSDSP" |
| 21 | | | USERVAR "CODEPAGE" VALUE "37" |
| 22 | | | USERVAR "CHARSET" VALUE "697" |
| 23 | | | USERVAR "KBDTYPE" VALUE "USB" |
| 24 | | | USERVAR "IBMSENDCONFREC" VALUE "YES" |
| 25 | | <- | IAC SE |
| 26 | | | FFFA2700 00555345 52014A53 54455645 |
| 27 | | | 4E530349 424D5253 45454401 04696CD0 |
| 28 | | | D7C41F81 0349424D 53554253 50570131 |
| 29 | | | 96A30203 3F5321FD 03444556 4E414D45 |
| 30 | | | 014A4546 46534453 5003434F 44455041 |
| 31 | | | 47450133 37034348 41525345 54013639 |
| 32 | | | 37034B42 44545950 45015553 4249424D |
| 33 | | | 53454E44 434F4E46 52454301 594553FF |
| 34 | | | F0 |
| 35 | | | |
| 36 | IAC SB TERMTYPE SEND | | |

```
IAC SE                                  ->
FFFA1801 FFF0
                                        <- IAC SB TERMTYPE IS IBM-3179-2 IAC SE
                                           FFFA1800 49424D2D 33313739 2D32FFF0
IAC DO EOR                              ->
FFFD19
                                        <- IAC WILL EOR
                                           FFFB19
IAC WILL EOR                            ->
FFFB19
                                        <- IAC DO EOR
                                           FFFD19
IAC DO BINARY                           ->
FFFD00
                                        <- IAC WILL BINARY
                                           FFFB00
IAC WILL BINARY                         ->
FFFB00
                                        <- IAC DO BINARY
                                           FFFD00
Display Confirmation Record             ->
004912A0 90000560 060020C0 003D0000
C9F9F0F2 D9E2F0F1 F0404040 D1C5C6C6
E2C4E2D7 40400000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00FFEF

RFC 1205 Data Stream                    ->
001112A0 00000400 000304F3 0005D970
00FFEF
```

Device name collision occurs when a Telnet client 40 sends the Telnet server 42 a virtual device name that it wants to use, but that device is already in use on the server 42. When this occurs, the Telnet server 42 sends a request to the client 40 asking it to try another device name. The environment option negotiation uses the USERVAR name of DEVNAME to communicate the virtual device name. Table 6 shows how the Telnet server 42 requests the Telnet client 40 to send a different DEVNAME when device name collision occurs, and is an example of how negotiations are done using environment variables, such as DEVNAME, USER,

CODEPAGE, CHARSET, and so forth. These are negotiations for various display session attributes which, according to the parent application, is enhanced to include IBMSENDCONFREC. In accordance with RFC 2877, the three fields, response code 114, system name 118 and device name 120, are the only useful fields that are returned, and those only for printer emulator sessions.

**Table 6 Negotiating Display Session Attributes**

```
   AS/400 Telnet server                Enhanced Telnet client
   --------------------------          -------------------------
   IAC SB NEW-ENVIRON SEND
   VAR USERVAR IAC SE          -->

   Server requests all environment variables be sent.

                                       IAC SB NEW-ENVIRON IS USERVAR
                                       "DEVNAME" VALUE "MYDEVICE1"
                                       USERVAR "xxxxx" VALUE "xxx"
                                       ...
                               <--     IAC SE

Client sends all environment variables, including DEVNAME.  Server tries
to select device MYDEVICE1.  If the device is already in use, server
requests DEVNAME be sent again.

   IAC SB NEW-ENVIRON SEND
   USERVAR "DEVNAME" IAC SE    -->

Server sends a request for a single environment variable: DEVNAME

                                       IAC SB NEW-ENVIRON IS USERVAR
                               <--     "DEVNAME" VALUE "MYDEVICE2" IAC SE

     Client sends one environment variable, calculating a new value of
MYDEVICE2.  If MYDEVICE2 is different from the last request, then server
tries to select device MYDEVICE2, else server disconnects client.  If
MYDEVICE2 is also in use, server will send DEVNAME request again, and
keep doing so until it receives a device that is not in use, or the same
device name twice in row.
```

Table 7 provides a detailed representation of the environment option negotiations in accordance with the present invention. In this example of a custom confirmation record, a Telnet client requesting the custom confirmation record is working in conjunction with a user exit program running on the server. The client can negotiate the environment variable "IBMSENDCUSTOMCONFREC" with ANY data as the value (not just a "YES" / "NO" value as with the regular confirmation record). It is then up to the user exit on the server to interpret this value and then send the appropriate information back to the client in the custom confirmation record. So the custom confirmation record contains the diagnostic information provided by the telnet server along with the custom information provided by the exit program.

In the example of Table 7, the client would like to know the interactive subsystem name this client job will be running in. It requests a custom confirmation record with a value of "INTERACTIVE SUBSYSTEM". Then the user exit program running on the server processes this value and sends back the interactive subsystem name, "SALES001" (hex '53 41 4C 45 53 30 30 31') in custom confirmation record.

**Table 7: TN5250E Environment Option Negotiations**

| Telnet Server | | Telnet Client |
|---|---|---|
| IAC DO NEW-ENVIRON<br>FFFD27 | -> | |
| | <- | IAC WILL NEW-ENVIRON<br>FFFB27 |
| IAC DO TERMTYPE<br>FFFD18 | -> | |
| | <- | IAC WILL TERMTYPE<br>FFFB18 |
| IAC SB NEW-ENVIRON SEND<br>USERVAR "IBMRSEEDxxxxxxxx"<br>USERVAR "IBMSUBSPW"<br>VAR USERVAR IAC SE<br>FFFA2701 0349424D 52534545 447D68B9<br>2BE04E04 040003FF F0 | -> | |
| | | IAC SB NEW-ENVIRON IS<br>VAR "USER" VALUE "JSTEVENS"<br>USERVAR "IBMRSEED" VALUE<br>USERVAR "IBMSUBSPW" VALUE<br>"YYYYYYYY"<br>USERVAR "DEVNAME" VALUE<br>"JEFFSDSP"<br>USERVAR "CODEPAGE" VALUE "37"<br>USERVAR "CHARSET" VALUE "697"<br>USERVAR "KBDTYPE" VALUE "USB"<br>USERVAR "IBMSENDCUSTOMCONFREC"<br>VALUE "INTERACTIVE SUBSYSTEM" |
| | <- | IAC SE<br>FFFA2700 00555345 52014A53 54455645<br>4E530349 424D5253 45454401 04696CD0<br>D7C41F81 0349424D 53554253 50570131<br>96A30203 3F5321FD 03444556 4E414D45<br>014A4546 46534453 5003434F 44455041<br>47450133 37034348 41525345 54013639<br>37034B42 44545950 45015553 42034942<br>4D53454E 44435553 544F4D43 4F4E4652<br>45430149 4E544552 41435449 56452053<br>55425359 5354454D FFF0 |
| IAC SB TERMTYPE SEND<br>IAC SE<br>FFFA1801 FFF0 | -> | |
| | <- | IAC SB TERMTYPE IS IBM-3179-2 IAC<br>SE<br>FFFA1800 49424D2D 33313739 2D32FFF0 |
| IAC DO EOR<br>FFFD19 | -> | |

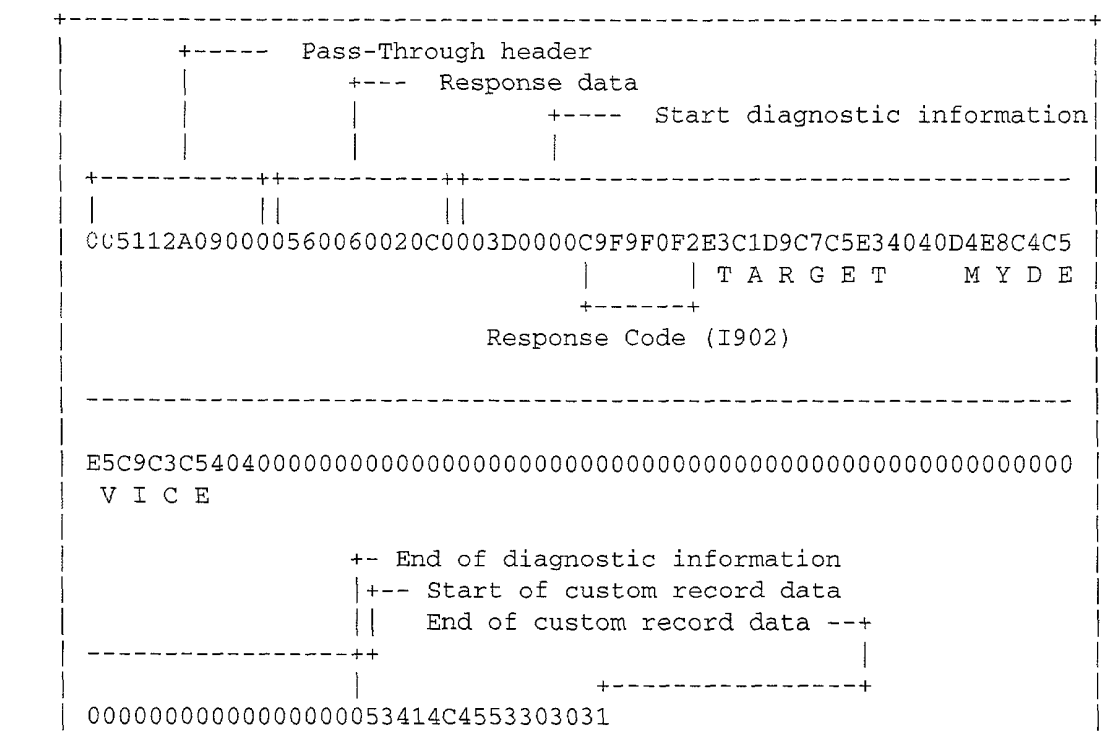
```
                                        <- IAC WILL EOR
                                           FFFB19
IAC WILL EOR                            ->
FFFB19
                                        <- IAC DO EOR
                                           FFFD19
IAC DO BINARY                           ->
FFFD00
                                        <- IAC WILL BINARY
                                           FFFB00
IAC WILL BINARY                         ->
FFFB00
                                        <- IAC DO BINARY
                                           FFFD00
Display Confirmation Record             ->
005112A0 90000560 060020C0 003D0000
C9F9F0F2 D9E2F0F1 F0404040 D1C5C6C6
E2C4E2D7 40400000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 0053414C 45533030
31FFEF

RFC 1205 Data Stream                    ->
001112A0 00000400 000304F3 0005D970
00FFEF
```

Confirmation record 122 has a length member 108, length pass-through data, that in accordance with the present invention, allows extension of data being sent to include any custom information that an exit program may want to insert. This custom information 124 is appended to the end of the confirmation record, thus ensuring backwards compatibility with confirmation records 120 which do not include a custom record data field 124. In accordance with the preferred embodiment of the invention, a new environment variable, for example, "IBMSENDCUSTOMCONFREC" with a value of "YES" may be sent instead of "IBMSENDCONFREC". In response, all of the old confirmation record 120 data is

returned along with any new information in custom record data field 124. This custom data 124 can be anything an application wishes to include, and can be in any format desired. It will be up to client emulator, for example, to interpret any data inserted in this custom area 124. Table 8 illustrates this concept, using the custom confirmation record presented in Table 7. The length data 108 is adjusted from '49'X to '51'X to account for the additional custom data.

**Table 8 Example Custom Display Confirmation Record Layout** for a Success or Error Response Record




- '0051'X = Length pass-through data, including this length field
- '12A0'X = GDS LU6.2 header
- '900005600600 20C0003D0000'X = Fixed value fields
- 'C9F9F0F2'X = Response Code (I902)
- 'E3C1D9C7C5E34040'X = System Name (TARGET)
- 'D4E8C4C5E5C9C3C54040'X = Object Name (MYDEVICE)
- '53414C4553303031'X = Custom Data (SALES001)

## Advantages over the Prior Art

It is an advantage of the invention that there is provided an improved system and method for establishing a client/server connection.

It is a further advantage of the invention that there is provided an improved system and method for negotiating a client/server connection in a connection-oriented protocol.

It is a further advantage of the invention that there is provided a system and method allowing customers to exploit custom solutions between clients and servers.

It is a further advantage of the invention that there is provided a system and method for enabling client emulators to request that custom information be returned by servers.

It is a further advantage of the invention that there is provided a system and method for exploiting confirmation records technology to enable clients to receive custom information from servers during session connection.

## Alternative Embodiments

It will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the

invention. In particular, it is within the scope of the invention to provide a computer program product or program element, or a program storage or memory device such as a solid or fluid transmission medium, magnetic or optical wire, tape or disc, or the like, for storing signals readable by a machine, for controlling the operation of a computer according to the method of the invention and/or to structure its components in accordance with the system of the invention.

Further, each step of the method may be executed on any general computer, such as an IBM System 390 (z Series), AS/400 (i Series), PC (x Series), p Series, or the like and pursuant to one or more, or a part of one or more, program elements, modules or objects generated from any programming language, such as C++, Java, Pl/1, Fortran or the like. And still further, each said step, or a file or object or the like implementing each said step, may be executed by special purpose hardware or a circuit module designed for that purpose.

While the preferred embodiment of the invention has been described primarily with respect to a Telnet environment or protocol, in a broader sense it is applicable to any connection oriented client/server protocol, such as a TCP/IP family of applications. Such protocols may make use of a confirmation record, served in accordance with the preferred embodiments of the present invention, confirming the status or other attributes associated with an actual connection. An example of such a protocol is the file transfer protocol (FTP), in which a connection is initiated and held for the duration of a file transfer. Telnet initiates and holds the connection for the duration of the dialogue between the attaching client emulator that initiates the connection to a targeted host server and its application.

Accordingly, the scope of protection of this invention is limited only by the following claims and their equivalents.